Learning a Machine for the Decision in a Partially Observable Markov Universe

Frédéric Dambreville

Délégation Générale pour l'Armement, DGA/CTA/DT/GIP

16 Bis, Avenue Prieur de la Côte d'Or

F 94114, France

Email: `Frederic.DAMBREVILLE@dga.defense.gouv.fr`

February 1, 2008

Abstract

In this paper, we are interested in optimal decisions in a partially observable Markov universe. Our viewpoint departs from the dynamic programming viewpoint: we are directly approximating an optimal strategic tree depending on the observation. This approximation is made by means of a parameterized probabilistic law. In this paper, a particular family of hidden Markov models, with input *and* output, is considered as a learning framework. A method for optimizing the parameters of these HMMs is proposed and applied. This optimization method is based on the cross-entropic principle.

Keywords: Control, MDP/POMDP, Hierarchical HMM, Bayesian Networks, Cross-Entropy

Notations. Some specific notations are used in this document.

- The variables x, y, z and m are used for the action, observation, world state and machine memory,

- The time t is starting from stage 1 to the maximal stage T. Variables with subscript outside this scope are synonymous to $\emptyset$. For example, $\prod_{t=1}^{T} \pi(x_t|x_{t-1})$ means $\pi(x_1|\emptyset) \prod_{t=2}^{T} \pi(x_t|x_{t-1})$, *ie.* a Markov chain. A similar principle is used for the *level* supscript λ in the definition of hierarchical HMM,

- Braces used with subscripts, *eg.* $\{\}_{t=1}^{T}$, only have a grammatical meaning. More precisely, it means that the symbols inside the braces are duplicated and concatenated according to the subscript. For example, $\{f_k(\}_{k=1}^{3}x\{)\}_{k=3}^{1}$ means $f_1(f_2(f_3(x)))$ and $\{x_k\}_{k=t}^{T}$ means $\prod_{k=t}^{T} x_k$,

- The generic notation for a probability is P. However, the functions p, π and h denote some specific components of the probability. p is the law of the observation y and state z conditionally to the action x. π is a stochastic policy, *ie.* a law of the action conditionally to the observation. h is an approximation of π by a HMM family. The hidden state of h is defined as the machine memory m.

1   Introduction

There are different degrees of difficulty in planning and control problems. In most problems, the planner have to start from a given state and terminate in a required final state. There are several transition rules, which condition the sequence of decision. For example, a robot may be required to move from room A, starting state, to room B, final state; its action could be *go forward*, *turn right* or *turn left*, and it cannot cross a wall; these are the conditions over the decision. A first degree in the difficulty is to find at least one solution for the planning. When the states are only partially known or the actions are not deterministic, the difficulty is quite enhanced: the planner has to take into account the various observations. Now, the problem becomes much more complex, when this planning is required to be optimal or near-optimal. For example, find the shortest trajectory which moves the robot from room A to room B. There are again different degrees in the difficulty, depending on the problem to be deterministic or not, depending on the model of the future observations. In the particular case of a Markovian problem with the full observation hypothesis, the dynamic programming principle could be efficiently applied (Markov Decision Process theory/MDP). This solution has been extended to the case of partial observation (Partially Observable Markov Decision Process/POMDP), but this solution is generally not practicable, owing to the huge dimension of the variables.

In this paper, we are interested in optimal planning with partial observation. A Markovian hypothesis is made. Our viewpoint departs from the dynamic programming viewpoint: we are directly approximating an optimal strategic tree depending on the observation. This approximation is made by means of a parameterized conditional probabilistic law, *ie.* actions depending on observations. The main problems are the choice of the parameterized laws family and the learning of the optimal parameters. A particular family of (hierarchical) hidden Markov models, with input *and* output, has been used as the learning framework. A method for optimizing the parameters of these HHMMs is proposed and applied. This optimization method is based on the cross-entropic principle[1]. The resulting parameterized law could be seen as a *Virtual Machine* which could generate near-optimal dynamic strategies for any drawing of the problem.

The next section introduces some formalism and gives a quick description of the MDP/POMDP problems. It is recalled that the Dynamic Programming method could solve these problems, but that this solution is intractable for an actual POMDP problem. It is then proposed a new near-optimal planning method, based on the direct approximation of the optimal decision tree. The third section introduces the family of Hierarchical Hidden Markov Models. A particular sub-family of HHMMs is proposed as a candidate for the approximation of decision trees. The fourth section describes the method for optimizing the parameters of the HHMM, in order to approximate the optimal decision tree for the POMDP problem. The cross-entropy method is described and applied. The fifth section gives an example of application. The paper is then concluded.

2   MDP and POMDP

It is assumed that a subject is acting in a given world with a given purpose or mission. The goal is to optimize the accomplishment of this mission. As a framework for the optimization, a prior model is hypothetized for the world (world state and observation depending on the action of the subject) and for the mission evaluation (a function depending on world state, observation and action). In the next paragraphs, a Markovian modelling of the world

3

Figure 1: A Markovian World
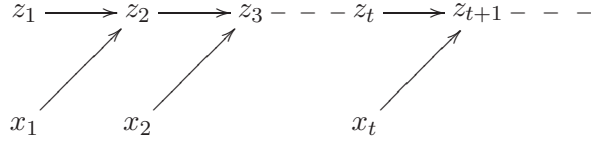
and a recursive modelling of the evaluation are considered.

The world. Our modelling of the world is based on the MDP/POMDP formalism, *ie.* (Partially Observable) Markov Decision Process. In the MDP or POMDP formalism, the world is considered as a Markov chain conditioned by the past action of the subject. Assume that the world is characterized by a *state* variable z and that the action is described by a variable x. Assume that the time is starting from 1, and denote respectively z_t and x_t the world state and the action for the time t. The law of z with respect to the past actions x thus verifies the property of Markov:

$$P(z_t|z_{1:t-1}, x_{1:t-1}) = p(z_t|z_{t-1}, x_{t-1}) ,$$

where the notation $z_{1:t}$ means $z_1, \ldots, z_t$, and the notation z_0, *ie.* before starting time, means $\emptyset$. This Markov chain is thus characterized by two functions: an initial law $p(z_1|\emptyset)$ and a transition law $p(z_t|z_{t-1}, x_{t-1})$ for $t \geq 2$. This distinction is often omitted in this paper, but it is implied. The law of $z|x$ is represented graphically by the Bayesian Network of figure 1. In this description, the arrows indicate the dependency between variables. For example, $z_t \to z_{t+1} \leftarrow x_t$ means that the law of z_{t+1} is defined conditionally to the past variables x_t and z_t. Such Bayesian Network representations will be used several times in this paper for their pedagogic skill.

The observation. MDP and POMDP assume that the world is observed during the process.

In MDP, the world is fully observed; *the observation at time t is the world state z_t.*

In POMDP, the world is partially observed; *the observation at time t is*

Figure 2: A Hidden Markov Model (with control)
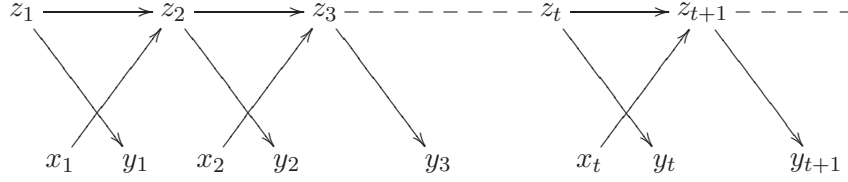
denoted y_t. The variable of observation y_t is only depending on the current world state z_t:

$$P(y_t|z_{1:t}, y_{1:t-1}, x_{1:t}) = p(y_t|z_t) .$$

The variables $y, z|x$ constitute a *Hidden Markov Model* (with control). This HMM is represented in figure 2.

Evaluation and optimal planning. The previous paragraphs have built a modelization of the world, of the actions and of the observations. We are now giving a characterization of the mission to be accomplished. *The mission is limited in time.* Let T be this maximum time. In the most generality, the mission is evaluated by a function $V(x_{1:T}, y_{1:T}, z_{1:T})$ defined on the trajectories of x, y, z ($V(x_{1:T}, z_{1:T})$ in the case of a MDP). Typically, the function V could be used for computing the time needed for the mission accomplishment. The purpose is to construct an optimal decision tree $x(obs)$ depending on the observation *obs* in order to maximize the *mean evaluation*. This is a dynamic optimization problem, since the actions depend on the previous observations. This problem is expressed differently for MDP and POMDP:

MDP. $obs_t = z_t$. Optimize $x_t(z_{1:t-1})|_{1 \leq t \leq T}$:

Find $\quad x_O \in \arg\max_x \sum_{z_{1:T}} V(x_t(z_{1:t-1})|_{1 \leq t \leq T}, z_{1:T}) \prod_{t=1}^{T} p(z_t|z_{t-1}, x_{t-1}(z_{1:t-2})) .$
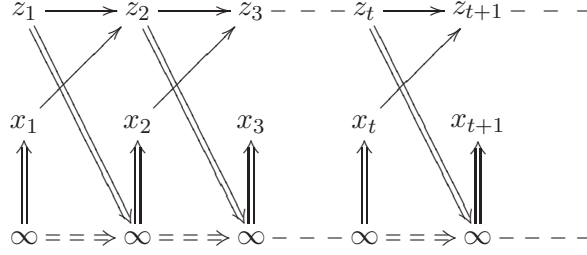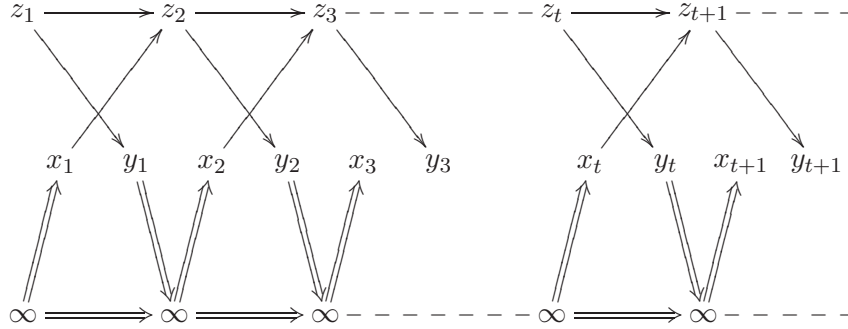
Figure 3: MDP planning

Figure 4: POMDP planning

POMDP. $obs_t = y_t$. Optimize $x_t(y_{1:t-1})|_{1 \leq t \leq T}$:

$$\text{Find} \quad x_O \in \arg \max_x \sum_{y_{1:T}} \sum_{z_{1:T}} V(x_t(y_{1:t-1})|_{1 \leq t \leq T}, y_{1:T}, z_{1:T})$$
$$\prod_{t=1}^T p(y_t|z_t) \, p(z_t|z_{t-1}, x_{t-1}(y_{1:t-2})) \,. \tag{1}$$

These optimizations are schematized in figures 3 and 4. In these figures, the double arrows $\Rightarrow$ characterize the variables to be optimized. More precisely, these arrows describe the flow of information between the observations and the actions. *The cells denoted ∞ are making decisions and transmitting all the received and generated information* (including the actions). This architecture illustrates that planning with observation is an *in(de)finite-memory* problem : the decision depends on the whole past observations. But in the particular case of MDP, the problem is finite memory : the dashed arrows of figure 3 may be removed resulting in the finite memory BN of
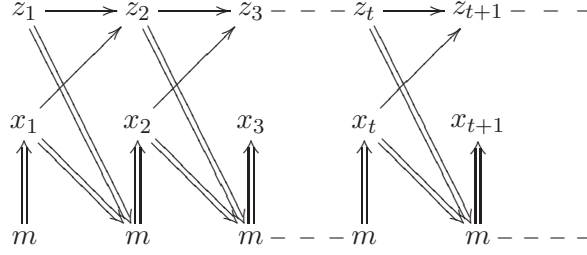
6

Figure 5: MDP planning with finite memory

figure 5 (memory of the past decision is maintained).

This "finiteness" is an ingredient which makes the MDP problems rather easily solvable by means of the dynamic programming method. DP methods are introduced in section 2.1. As a result, DP produces the root $x_{O,1}$ of the optimal decision tree, which is a sufficient tool for the decision in an actual open-loop planning process. More precisely, the whole process for optimizing the actual mission takes the form:

1. Initialize the MDP/POMDP priors of the mission,

2. Compute the theoretical optimal decision root $x_{O,1}$,

3. The decision $x_{O,1}$ is applied to the actual world,

4. Make actual observations and update the prior; in particular, the time is forwarded by 1,

5. Repeat from step 2 until accomplishment of the mission.

2.1 Dynamic Programming method

This presentation is extremely simplified. More detailed references are available[2][3][4]. The Dynamic Programming method works for an additive evaluation $V(x_{1:T}, z_{1:T}) = \sum_{t=1}^{T} V_t(x_t, z_t)$. In this simplified presentation, a final evaluation is assumed, *ie.* $V(x_{1:T}, z_{1:T}) = V_T(x_T, z_T)$, but the principle is the same.

The MDP case. To be computed:

$$x_{O,1} \in \arg_{x_1} \max_x \sum_{z_{1:T}} V_T(x_T(z_{1:T-1}), z_T) \prod_{t=1}^{T} p(z_t | z_{t-1}, x_{t-1}(z_{1:t-2})) \, .$$

This computation is easily refined:

$$x_{O,1} \in \arg_{x_1} \left\{ \max_{x_t} \sum_{z_t} p(z_t | z_{t-1}, x_{t-1}) \right\}_{t=1}^{T} V_T(x_T, z_T) \, . \qquad (2)$$

This factorization is deeply related to the finite memory property of the MDP. More precisely, the "optimal" evaluation for stage t, denoted W_t, may be computed recursively, as a direct result of (2) :

$$\begin{cases} W_T(x_T, z_T) = V_T(x_T, z_T) \, , \\ W_t(x_t, z_t) = \max_{x_{t+1}} \sum_{z_{t+1}} p(z_{t+1} | z_t, x_t) W_{t+1}(x_{t+1}, z_{t+1}) \, . \end{cases}$$

Associated to these evaluations are defined the optimal strategies $x_{O,t}$ for the stage t :

$$x_{O,t} \in \arg \max_{x_t} \sum_{z_t} p(z_t | z_{t-1}, x_{t-1}) W_t(x_t, z_t) \, .$$

In particular: $x_{O,1} \in \arg \max_{x_1} \sum_{z_1} p(z_1 | \emptyset) W_1(x_1, z_1) \, .$

It happens that the optimal strategy for stage t is only depending on the previous action and state x_{t-1}, z_{t-1} . This is exactly what have been intuited from figure 5 as a finite memory property.

This method, known as the dynamic programming, thus relies on the computation of backwards functions defined over the variables x_t, z_t . The DP methods have been extended to POMDP problems, but this time the recursive functions are much more intricated and almost intractable.

The POMDP case. To be computed:

$$x_{O,1} \in \arg_{x_1} \max_x \sum_{y_{1:T}} \sum_{z_{1:T}} V_T(x_T(z_{1:T-1}), y_T, z_T)$$

$$\prod_{t=1}^{T} p(y_t | z_t) \, p(z_t | z_{t-1}, x_{t-1}(y_{1:t-2})) \, .$$

8

The previous factorization works partially:

$$x_{O,1} \in \arg_{x_1} \left\{ \max_{x_t} \sum_{y_t} \right\}_{t=1}^{T} \sum_{z_{1:T}} V_T(x_T, y_T, z_T) \prod_{t=1}^{T} p(y_t|z_t)\, p(z_t|z_{t-1}, x_{t-1}) \;.$$

But this factorization is incomplete and the previous recursion is not possible anymore. Still, there is an answer to this problem by means of a dynamic programming method. But the recursion cannot rely on the variables x_t, y_t (last action and observation) only: these variables contain insufficient informations to predict the future. It is necessary to transmit the probabilistic *belief* over z_t estimated from the whole past actions and observations. Denote $b_t(z_t)$ the belief over z_t. The solution of the POMDP is constructed recursively:

$$
\begin{cases}
W_T(b_T) = \max_{x_T} \sum_{y_T} \sum_{z_T} b_T(z_T) p(y_T|z_T) V_T(x_T, y_T, z_T) \;; \\[2mm]
\beta_{t+1}(z_{t+1}|x_t, y_t, b_t) = \sum_{z_t} b_t(z_t) p(y_t|z_t) p(z_{t+1}|z_t, x_t) \;, \\[2mm]
W_t(b_t) = \max_{x_t} \sum_{y_t} W_{t+1}(\beta_{t+1}(\cdot|x_t, y_t, b_t)) \;; \\[2mm]
\beta_2(z_2|x_1, y_1) = \sum_{z_1} p(z_1|\emptyset) p(y_1|z_1) p(z_2|x_1, z_1) \;, \\[2mm]
x_{O,1} \in \arg \max_{x_1} \sum_{y_1} W_2(\beta_2(\cdot|x_1, y_1)) \;.
\end{cases}
$$

This recursive construction of W_t is very difficult, since the variable b_t is continuous and has a very high dimension. For example, if the state z_t is the position of a target in a discrete space of 20×20 cells (the same space as in the example[1] of section 5), this dimension is 399. Of course, there are many possible refinements and approximations, but even so, the DP method becomes easily intractable.

This particular difficulty for solving POMDP problems was illustrated by the figures 3 and 4: MDPs are fundamentally more simple than POMDPs. Indeed, since the world state z is a Markov chain, the knowledge of z_t together with the last action x_t is sufficient to predict all the future (figure 5). On the other hand, the knowledge of y_t and of x_t is not sufficient to predict the future of the Markov chain z.

---

[1]In this example the dimension is even more, *ie.* $4^2 400^3 - 1$.

2.2 Direct approximation of the decision tree

While the classical DP method generally fails solving the POMDP optimization, some authors have investigated approximations of the solution. Our work investigates the direct approximation of the decision tree.

In an optimization problem like (1), the value to be optimized, x_O, is a *deterministic* object. In this precise case, x_O is a tree of decision, that is a function which maps to a decision x_t from any sequence of observation $y_{1:t-1}$. It is possible however to have a probabilistic viewpoint.[2] Then the problem is equivalent to finding $x, y \mapsto \pi(x|y)$, a probabilistic law of actions conditionally to the *past* observations, which maximizes the mean evaluation:

$$V(\pi) = \sum_{x_{1:T}} \sum_{y_{1:T}} \sum_{z_{1:T}} \prod_{t=1}^{T} \pi(x_t|x_{1:t-1}, y_{1:t-1})$$
$$\times P(y_{1:T}, z_{1:T}|x_{1:T})\, V(x_{1:T}, y_{1:T}, z_{1:T})\,,$$

where:

$$P(y_{1:T}, z_{1:T}|x_{1:T}) = \prod_{t=1}^{T} p(y_t|z_t)\, p(z_t|z_{t-1}, x_{t-1})\,.$$

If the solution is optimal, there will not be a great difference with the deterministic case: when the solution x_O is unique, the optimal law π_O is a dirac on x_O. But things change when approximating π. Now, why using a probability to approximate the optimal strategy? The main point is that probabilistic models seem more suitable for approximation, than for example deterministic decision trees. The second point is that we are sure to approximate continuously: indeed, $\pi \mapsto V(\pi)$ is continuous. There is a third point, which is more "philosophic". Considering the figure 4, it appears clearly that it is symmetric: the arrows $\Rightarrow$ and $\rightarrow$ play a quite similar role. In the deterministic viewpoint, this is just a coincidence. But if we are considering probabilistic strategies, the arrows $\Rightarrow$ in the Bayesian Network of figure 4 are constituting a HMM controlled by the observation, and with infinite memory. Of course, a natural approximation of a HMM with infinite memory is a HMM with finite memory! Then replacing the infinite-memory states ∞ by a finite-memory variable m, a perfectly symmetric problem is

---

[2]Such probabilistic viewpoint is more common, in fact necessary, in game problems.

Figure 6: Finite-memory planning approximation
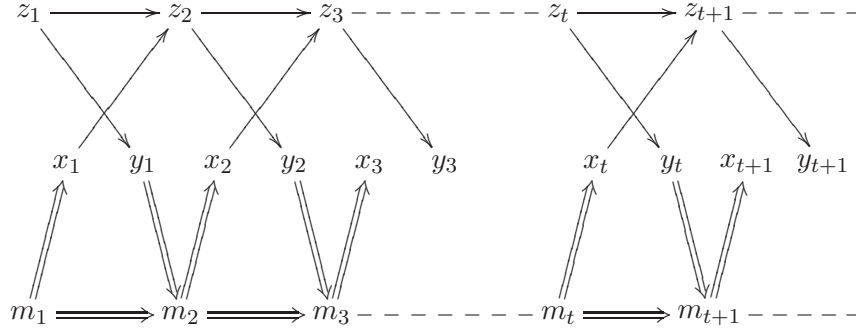
obtained in figure 6; perhaps some kind of duality relation. Well, aestheticism is not the unique interest of such approximation. The most interesting point is that there are many methods which apply on HMM. Particularly for learning and optimizing parameters. Then, the approach developped in this paper is quite general and can be split up into two points:

- Define a family of parameterized HMMs $\mathcal{H}$,

- Optimize the parameters of the HMM in order to maximize the mean evaluation:

$$\text{Find} \quad h_O \in \arg\max_{h \in \mathcal{H}} V(h) .$$

The first point is discussed in the next section, where a sub-class of hierarchical HMMs is defined. The second point is discussed in the following section, which explains a cross-entropic method for optimizing the parameters.

3   Hierarchical Hidden Markov Model

In this work, a significant investment was dedicated to the construction of a "good" parameterized probabilitic family. In order to investigate future complex problems, our choice has focused on hierarchical HMM models. These models are inspired from biology: to solve a complex problem, factorize it and make decisions in a hierarchical fashion. Low hierarchies manipulate low level informations and actions, making short-term decisions. High hierarchies manipulate high level informations and actions (uncertainty is less),

11

making long-term decisions.

The implementation of such hierarchical models have not been completely investigated yet; a rather simple HHMM has been implemented, and our example tests just needed few hierarchies. Still, this section is deliberately discussing about HHMM with more hindsight.

3.1 Generality

This section gives some general references about HHMMs *without control*. However, HHMM have been used yet in control problems. In [5] a HHMM modelling of the world is used in a POMDP problem, but the control (*ie.* the action) is not propagated within the HHMM. In our models, actions and observations are both propagated within the same HHMM structure. Thus, the model of controlled HHMM defined here is not related to any existing work, the author knows. But it is interesting to introduce this section with some results about uncontrolled HHMM.

A hierarchical hidden Markov model (HHMM) may be defined as a HMM which output is either a hierarchical HMM or an actual output. A HHMM could also be considered as a hierarchy of *stochastic* processes calling sub-processes. It is noteworthy that the length of a sub-process call depends on the sub-process law. This length is variable.

Simple HMM have been variously applied, for example in speech or hand-writing recognition, in robotic, etc. However there are still few applications of hierarchical HMM[5][6], although it is probable that HHMM should allow a more abstract representation of the processes.

In [6], *Fine, Singer and Tishby* applied HHMM to *identify combinations of letters in handwriting*. The main difficulty was to derive appropriate Viterbi and Baum-Welch algorithms. The Complexity of *Fine and al* methods was about $T^3 Q^\Lambda$, where T is the length of the samples, Λ is the number of *levels* of the HHMM tree and Q is the number of states per levels of the HHMM. Such required computation times are obviously difficult to apply; for most samples T is too big.

In their work [7], *Murphy and Paskin* have shown how HHMM could be

12

Figure 7: Model of a Hierarchical HMM

$\bullet$ / $\circ$ / $\triangledown$ = information / information+boolean / ouput
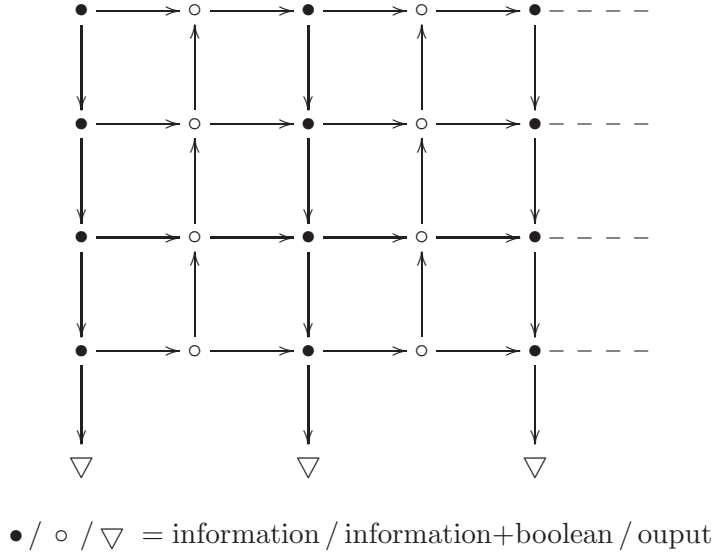
interpreted as a particular 2−dimension dynamic Bayesian Network sized $\Lambda \times T$, and derived algorithms of complexity $TQ^{2\Lambda}$. This is much better, but remains exponential with Λ. The number of levels is thus strongly limited. Although *Murphy and Paskin* did not express this property explicitly, it is easy to show that their Bayesian Network has the Markov property both in time and in level. In fact, a hierarchical HMM could be interpreted by a Bayesian Network as described in figure 7 (refer to appendix A). A hierarchical HMM is thus clearly a HMM with vectorial states. However, these states should contains some boolean components which are used to define the sub-process call and return[7] (also refer to appendix A). Moreover, there is a downward *and upward* exchange of information between the levels, in a hierarchical HMM.

The next section introduces a model of controlled HHMM , which is (freely) inspired from the BN of figure 7 .

3.2   Definition of a controlled model

A natural extension of the BN of figure 7 is obtained by adding an input door to each upward column of the BN (then completing it), as described in

13

Figure 8: Model of a controlled Hierarchical HMM

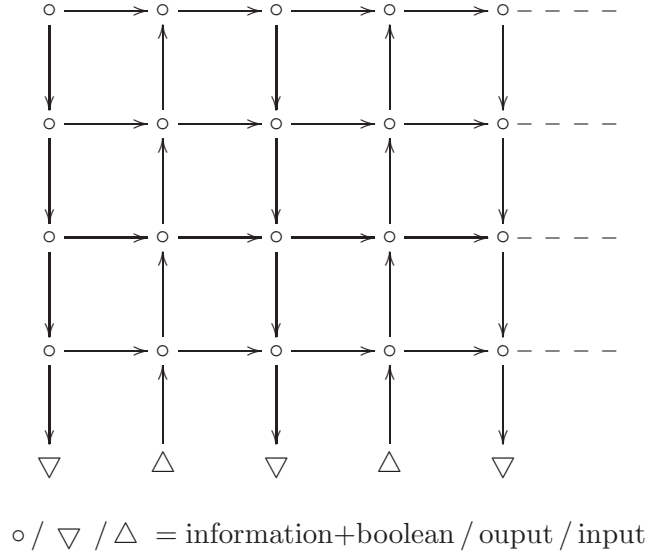$\circ / \bigtriangledown / \bigtriangleup$ = information+boolean / ouput / input

figure 8; moreover, boolean components are then needed in all hidden cells, so that the whole memory is required to be discrete or semi-continuous. Then, an alternance of output and input is obtained, which exactly matches our problem. This is probably a very good model and future works should investigate this solution. But for historical reasons in our work, another model, similar, has been chosen for now...

In the HHMM family which has been implemented, each memory state receives an information from the current upper-level state and the previous lower-level state. This model guarantees a propagation of the information between the several levels of the hierarchy, but this propagation is slower than in the case of figure 8. More precisely, this HHMM family being denoted $\mathcal{H}$, any hhmm $h \in \mathcal{H}$ takes the form:

$$ h(x|y) = \sum_{m_{1:T}^{1:\Lambda} \in M^{1:\Lambda}} \prod_{t=1}^{T} h^0(x_t|m_t^1) h^1(m_t^1|y_{t-1}, m_t^2) \prod_{\lambda=2}^{\Lambda} h^\lambda(m_t^\lambda|m_{t-1}^{\lambda-1}, m_t^{\lambda+1}) , $$

where m^λ is the variable, or memory, for the hidden level λ of the HHMM, M^λ is the set of possible states for the variable m^λ, and Λ is the number of levels of the HHMM. The law h is described graphically in figure 9. It is noteworthy that this model is equivalent to a simple HMM when $\Lambda = 2$.

Figure 9: HHMM model for the planning
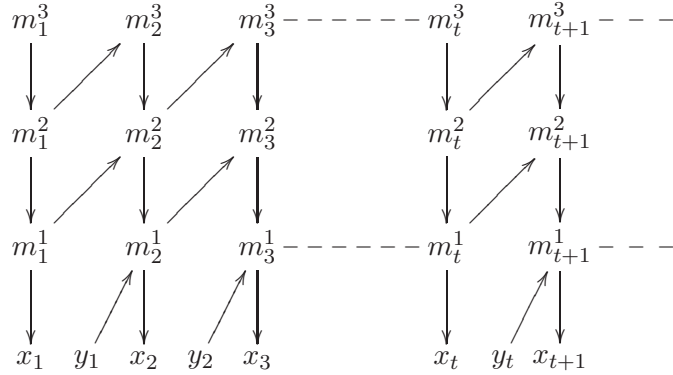
And when $\Lambda = 1$, the law h just maps the immediate observation to action, without any memory of the past observations.

For any $h \in \mathcal{H}$, define $P[h]$ the complete probabilistic law of the system Universe/Planner:

$$P[h](x, y, z, m) = P(y_{1:T}, z_{1:T}|x_{1:T}) \prod_{t=1}^{T} h^0(x_t|m_t^1)$$
$$\times h^1(m_t^1|y_{t-1}, m_t^2) \prod_{\lambda=2}^{\Lambda} h^\lambda(m_t^\lambda|m_{t-1}^{\lambda-1}, m_t^{\lambda+1}) .$$

Our *approximated* planning problem reduces to find the near-optimal strategy $h_O \in \mathcal{H}$ such that:

$$h_O \in \arg\max_{h \in \mathcal{H}} \sum_{x_{1:T}} \sum_{y_{1:T}} \sum_{z_{1:T}} \sum_{m_{1:T}^{1:\Lambda}} P[h](x, y, z, m) \, V(x_{1:T}, y_{1:T}, z_{1:T}) .$$

A solution to this problem, by means of the cross-entropy method, is proposed in the next section.

4 Cross-entropic optimization of h

The reader interested in CE methods should refer to the *tutorial on the CE method*[1]. CE algorithms were first dedicated to estimating the probability of rare events. A slight change of the basic algorithm made it also good for optimization. In their new article[8], Homem-de-Mello and Rubinstein

15

have given some results about the global convergence. In order to ensure such convergence, some refinements are introduced particularly about the selective rate.

This presentation is restricted to the CE optimization method. The new improvements of the CE algorithm proposed in [8] have not been implemented, but the algorithm has been seen to work properly. For this reason, this paper does not deal with the choice of the selective rate.

4.1   General CE algorithm for the optimization

The Cross Entropy algorithm repeats until convergence the three successive phases:

1. Generate samples of random data according to a parameterized random mechanism,

2. Select the best samples according to an evaluation criterion,

3. Update the parameters of the random mechanism, on the basis of the selected samples.

In the particular case of CE, the update in phase 3 is obtained by minimizing the Kullback-Leibler distance, or cross entropy, between the updated random mechanism and the selected samples. The next paragraphs describe on a theoretical example how such method can be used in an optimization problem.

Formalism.   Let be given a function $x \mapsto f(x)$; this function is easily computable. The value $f(x)$ has to be maximized, by optimizing the choice of $x \in X$. The function f will be the evaluation criterion.

Now let be given a family of probabilistic laws, $P_\sigma|_{\sigma \in \Sigma}$, applying on the variable x. The family P is the parameterized random mechanism. The variable x is the random data.

Let $\rho \in]0, 1[$ be a selective rate. The CE algorithm for (x, f, P) follows the synopsis :

1. Initialize $\sigma \in \Sigma$,

2. Generate N samples x_n according to P_σ,

3. Select the ρN best samples according to the evaluation criterion f,

4. Update σ as a minimizer of the cross-entropy with the selected samples:

$$\sigma \in \arg \max_{\sigma \in \Sigma} \sum_{n \text{ selected}} \ln P_\sigma(x_n) \, ,$$

5. Repeat from step 2 until convergence.

This algorithm requires f to be easily computable.

Interpretation. The CE algorithm tightens the law P_σ around the maximizer of f. Then, when the probabilistic family P is well suited to the maximization of f, it becomes equivalent to find a maximizer for f or to optimize the parameter σ by means of the CE algorithm. The problem is to find a good family... Another issue is the criterion for deciding the convergence. Some answers are given in [8]. Now, it is outside the scope of this paper to investigate these questions precisely. Our criterion was to stop after a given threshold of successive *unsuccessful tries* and this very simple method have worked fine on our problem.

4.2 Application

Optimizing $h \in \mathcal{H}$ means tuning the parameter h in order to tighten the probability $P[h]$ around the optimal values for V. This is exactly solved by the *Cross-Entropy* optimization method. However, it is required that the evaluation function V is easily computable. Typically, the definition of V may be recursive, *eg.*:

$$V(x_{1:T}, y_{1:T}, z_{1:T}) = \{v_t(x_t, y_t, z_t, \}_{t=T}^2 \, v_1(x_1, y_1, z_1) \, \{)\}_{t=T}^2 \, .$$

Let the *selective rate* ρ be a positive number such that $\rho < 1$. The cross-entropy method for optimizing h follows the synopsis:

1. Initialize h. For example a flat h,

2. Make N tossing $\theta^n = (x^n, y^n, z^n, m^n)$ according to the law $P[h]$,

3. Choose the ρN best samples θ^n according to the sample evaluation $V(x_{1:T}^n, y_{1:T}^n, z_{1:T}^n)$. Denote S the set of the selected samples,

4. Update h as the minimizer of the cross-entropy with the selected samples:

$$h \in \arg\max_{h \in \mathcal{H}} \sum_{n \in S} \ln P[h](\theta^n) , \qquad (3)$$

5. Reiterate from step 2 until convergence.

For our HHMM model, the maximization (3) is solved by:

$$h^0(A|B) = \frac{\mathrm{card}\Big\{ n \in S\,, t \,/\, A = x_t^n \ \text{ and } B = m_t^{1,n} \Big\}}{\mathrm{card}\Big\{ n \in S\,, t \,/\, B = m_t^{1,n} \Big\}} ,$$

$$h^1(A|B,C) = \frac{\mathrm{card}\Big\{ n \in S\,, t \,/\, A = m_t^{1,n}\,, B = y_{t-1}^n \ \text{ and } C = m_t^{2,n} \Big\}}{\mathrm{card}\Big\{ n \in S\,, t \,/\, B = y_{t-1}^n \ \text{ and } C = m_t^{2,n} \Big\}} .$$

and for $2 \le \lambda \le \Lambda$,:

$$h^\lambda(A|B,C) = \frac{\mathrm{card}\Big\{ n \in S\,, t \,/\, A = m_t^{\lambda,n}\,, B = m_{t-1}^{\lambda-1,n} \ \text{ and } C = m_t^{\lambda+1,n} \Big\}}{\mathrm{card}\Big\{ n \in S\,, t \,/\, B = m_{t-1}^{\lambda-1,n} \ \text{ and } C = m_t^{\lambda+1,n} \Big\}} .$$

4.3 Bypassing the prior modelling

POMDPs have a main drawback, they require the priors about the world and the mission evaluation to be defined. Defining the mission evaluation is not a difficult task in general. But it is quite uneasy to define the HMM which is modelling the world. For example, what is the structure (memory requirement, time/space hierarchies) of this HMM? What are the transition laws? In many works, the structure is defined by the designer of the system. But generally, the transition parameters are specified by learning from the actual world (*eg.* Baum-Welch algorithm). At last, the two step are implemented:

- Learn priorly the parameters of the HMM associated to the actual world (Baum-Welch),

- Solve the POMDP for the instant play.

This is what is generally done when using the dynamic programming viewpoint. But there is something nice with the cross-entropic methods. Since the algorithm just needs samples, the *HMM is not required*. More precisely,

since $P[h] = P(y_{1:T}, z_{1:T}|x_{1:T})h(x_{1:T}, m_{1:T}|y_{1:T})$, the update property (3) may be factorized and the following simple result is derived:

$$h \in \arg\max_{h \in \mathcal{H}} \sum_{n \in S} \ln h(x_{1:T}^n, m_{1:T}^n|y_{1:T}^n) \,,$$

What is needed is a stochatic process, which, in association with the policy h, will build the samples. This process may just be the *actual world* itself! Then, optimizing h only requires time for experimental implementations:

1. Initialize h,

2. N samples $\theta^n = (x^n, y^n, m^n)$ and their respective evaluations V^n are obtained by repeating N times the following procedure:

   (a) Initialize $t = 1$,

   (b) Make a toss of x_t^n and m_t^n according to the law $h(x_t, m_t|m_{t-1}^n, y_{t-1}^n)$,

   (c) *Execute the action x_t^n in the actual world,*

   (d) *Make the actual measurement y_t^n,*

   (e) Set $t = t + 1$,

   (f) Repeat from step 2b until $t > T$,

   (g) Make an actual evaluation, denoted V_n, of the whole play (this evaluation depends on the actions, observations, and of the *actual hidden states*),

3. Choose the ρN best samples θ^n according to the sample evaluation V^n. Denote S the set of the selected samples,

4. Update h as the minimizer of the cross-entropy with the selected samples:

$$h \in \arg\max_{h \in \mathcal{H}} \sum_{n \in S} \ln h(x_{1:T}^n, m_{1:T}^n|y_{1:T}^n) \,,$$

5. Reiterate from step 2 until convergence.

This method avoids any construction of the world prior $P(y, z|x)$, but it requires many experimentations on the actual world. This is sometimes unworkable.

The next section presents an example of implementation of the algorithm described in section 4.2.

19

5 Implementation

The algorithm has been applied to a simplified target detection problem. At this time, the *open-loop* implementation has not been investigated yet.

5.1 Problem setting

A target R is moving in a lattice of 20×20 cells, *ie.* $[\![0, 19]\!]^2$. R is tracked by two mobiles, B and C, controlled by the planner. The coordinate of R, B and C at time t are respectively denoted (i_R^t, j_R^t), (i_B^t, j_B^t) and (i_C^t, j_C^t). B and C have a very limited information about the target position, and are maneuvering much slower:

• A move for B (respectively C) is either: *turn left, turn right, go forward, no move*. Consequently, there are $4 \times 4 = 16$ possible actions for the planner. These moves cannot be combined in a single turn. No diagonal forward: a mobile is either directed up, right, down or left,

• The mobiles are initially positioned in the down corners, *ie.* $i_B^1 = 0$, $j_B^1 = 19$ and $i_C^1 = 19$, $j_C^1 = 19$. The mobile are initially directed *downward*,

• B (respectively C) observes whether the target relative position is forward or not. More precisely:

    ∘ when B is directed upward, the mobile knows whether $j_R < j_B$ or not,

    ∘ when B is directed right, the mobile knows whether $i_R > i_B$ or not,

    ∘ when B is directed downward, the mobile knows whether $j_R > j_B$ or not,

    ∘ when B is directed left, the mobile knows whether $i_R < i_B$ or not,

• B (respectively C) knows whether its distance with the target is less than 3, *ie.* $d_\infty(B, R) < 3$, or not. The distance d_∞ is defined by:

$$d_\infty(B, R) = \max\{|i_B - i_R|, |j_B - j_R|\} .$$

At last, there are $2^4 = 16$ possible observations for the planner.

Several test cases have been considered. In case 1, the target R does not move. In any other case, the target R chooses stochastically its next position in its neighborhood. Any move is possible (up/down, left/right, diagonals, no move). The probability to choose a new position is proportional to the

sum of the squared distance from the mobiles:

$$
\begin{cases}
P(R^{t+1}|R^t) = 0 \text{ if } |i_R^{t+1} - i_R^t| > 1 \text{ or } |j_R^{t+1} - j_R^t| > 1 \,, \\
P(R_{t+1}|R_t) \propto (i_R^{t+1} - i_B^t)^2 + (j_R^{t+1} - j_B^t)^2 \\
\qquad\qquad\qquad +(i_R^{t+1} - i_C^t)^2 + (j_R^{t+1} - j_C^t)^2 \quad \text{else} \,.
\end{cases}
$$

This definition was intended to favorize escape moves: more great is a distance, more probable is the move. But in such summation, a short distance will be neglected compared to a long distance (this was initially a mistake in the modelling!). It is implied that a distant mobile will hide a nearby mobile. This "deluding" property is interesting and will induce actually two different kinds of strategy, whithin the learned machines.

The objective of the planner is to maintain the target sufficiently closed to at least one mobile (in this example, the distance between the target and a mobile is required to be not more than 3). More precisely, the evaluation function, V, is just counting the number of such "encounter":
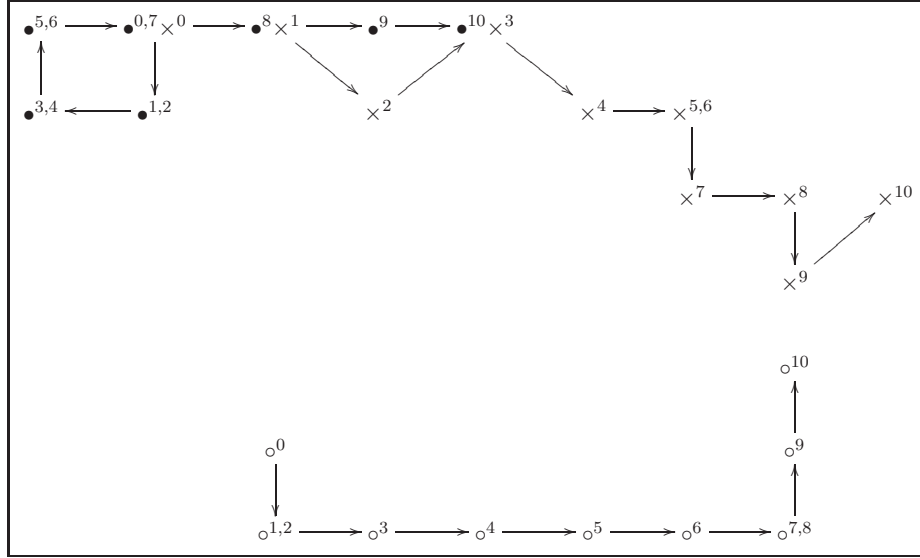
$$
V_0 = 0 \,; \quad V_t = V_{t-1}+1 \ \text{ if } d(B^t, R^t) \le 3 \text{ or } d(C^t, R^t) \le 3 \,; \quad V_t = V_{t-1} \ \text{ else.}
$$

The total number of turns is $T = 100$.

5.2   Results

Generality.   Like many stochastic algorithms, this algorithm needs some time for convergence. For the considered example, about two hours were needed for convergence (on a 2GHz PC); the selective rate was $\rho = 0.5$. This speed depends on the size of the HHMM model and on the convergence criterion. A weak and a strong criterion are used for deciding the convergence. Within the weak criterion, the algorithm is halted after 100 successive unsuccessful tries. Within the strong criterion, the algorithm is halted after 500 successive unsuccessful tries. Of course, the strong criterion computes a (slightly) better optimum than the weak criterion, but it needs time. Because of the many tested examples, the weak criterion has been the most used in particular for the big models. For the same HHMM model, the computed optimal values do not depend on the algorithmic instance (small variations result however from the stochastic nature of the algorithm): the convergence seems to be global.

Figure 10: Near-optimal control sequence

$\bullet^{5,6} \longrightarrow \bullet^{0,7} \times^0 \longrightarrow \bullet^8 \times^1 \longrightarrow \bullet^9 \longrightarrow \bullet^{10} \times^3$

$\bullet^{3,4} \longleftarrow \bullet^{1,2} \qquad \times^2 \qquad \times^4 \longrightarrow \times^{5,6}$

$\times^7 \longrightarrow \times^8 \qquad \times^{10}$

$\times^9$

$\circ^{10}$

$\circ^0 \qquad \circ^9$

$\circ^{1,2} \longrightarrow \circ^3 \longrightarrow \circ^4 \longrightarrow \circ^5 \longrightarrow \circ^6 \longrightarrow \circ^{7,8}$

$\times$ = target $\bullet$ = observer 1 $\circ$ = observer 2

Relative times are put in supscript

In the sequel, mean evaluations are rounded to the nearest integer. The presentation is made clearer. And owing to the small variations of this stochastic algorithm, more precision turns out to be irrelevant.

Case 1: R does not move. This example has been considered in order to test the algorithm. The position of the target is fixed in the center of the square space, *ie.* $i_R^1 = j_R^1 = 10$. It is recalled that the mobiles are initially directed downward. Then, the optimal strategy is known and its value is 85: the time needed to reach the target is 15, and no further move is needed. The learned h_O approximates the evaluation 84. The convergence is good.

Case 2: R is moving but the observation y is hidden. Initially, R is located within the 20×10 upper cells of the lattice (*ie.* $[\![\,0, 19\,]\!] \times [\![\,0, 9\,]\!]$), accordingly to a uniform probabilistic law. The computed optimal means evaluation is about 32. In this case, the mobiles tend to move towards the upper corners.

Case 3: R is moving and y is observed. Again, R^1 is located uniformly within the 20×10 upper cells of the lattice. The computed optimal means evaluation is about 69. This evaluation has been obtained from a large HHMM model ($\Lambda = 2$ with 256 states per level, *ie.* $\mathrm{card}(M^\lambda) = 256$) and with the strong criterion. However, somewhat smaller models should work as well.

Specific computations are now presented, depending on the number of levels Λ and the number of states per levels. For each case, the weak criterion has been used.

Subcase $\Lambda = 1$. For such model, the action x_t is constructed only from the immediate last observation y_{t-1}. The model does not keep any memory of the past observations. Then, only 16 states are sufficient to describe the hidden variable m_t^1, *ie.* $\mathrm{card}(M^1) = 16$. The resulting optimum is 54.

Subcase $\Lambda = 2$. This model is equivalent to a HMM and it is assumed that $\mathrm{card}(M^1) = \mathrm{card}(M^2)$. The following table gives the computed optimum for several choices of the set of states:

$\mathrm{card}(M^\lambda)$	16	32	64	256
Evaluation	65	66	67	67

It is noteworthy that the memory of the past observations allows better strategies than just immediate observations (case $\Lambda = 1$). Indeed, the evaluation jumps from 54 up to 67.

Subcases $\Lambda > 2$. A comparison of graduated hierarchic models, $1 \leq \Lambda \leq 4$, has been made. The first level contained 16 possible states, and the higger levels were restricted to 2 states:

hierarchic grade	$\Lambda = 1$	$\Lambda = 2$	$\Lambda = 3$	$\Lambda = 4$
$\mathrm{card}(M^\lambda)|_{\lambda=1}^{\Lambda}$	16	$16, 2$	$16, 2, 2$	$16, 2, 2, 2$

The test has been accomplished according to the weak criterion:

hierarchic grade	$\Lambda = 1$	$\Lambda = 2$	$\Lambda = 3$	$\Lambda = 4$
Evaluation (weak)	54	59	**56**	65

and the strong criterion:

hierarchic grade	$\Lambda = 1$	$\Lambda = 2$	$\Lambda = 3$	$\Lambda = 4$
Evaluation (strong)	55	61	64	66

23

It seems that a high hierarchic grade makes the convergence more difficult. This is particularly the case here for the grade $\Lambda = 3$, which failed under the weak criterion at only 56. However, the algorithm still works well by improving the convergence criterion.

It is interesting to make a comparison with the subcase $\Lambda = 2$ where $\mathrm{card}(M^1) = \mathrm{card}(M^2) = 16$. Under the weak criterion, the result for this HHMM was 65 as for the grade $\Lambda = 4$. However, the dimension of the law is quite different for the two models:

- $15 \times 16 + 15 \times 16 \times 16 + 15 \times 16 = 4320$ for the 2-level HHMM,

- $15 \times 16 + 15 \times 16 \times 2 + 1 \times 16 \times 2 + 1 \times 2 \times 2 + 1 \times 2 = 758$ for the 4-level HHMM.

This dimension is a rough characterization of the complexity of the model. It seems clear on these examples that the highly hierarchized models are more efficient than the weakly hierarchized models. And the problem considered here is quite simple. On complex problems, hierarchical models may be pre-eminent.

Global behavior.
The algorithm. The convergence speed is low at the beginning. After this initial stage, it improves greatly until it reaches a new "waiting" stage. This alternation of low speed and great speed stages have been noticed several times until an acceptable convergence. Nevertheless, the speed is globally decreasing with the time.

The near optimal policy. It is now discussed about the behaviour of the best found policy. This HHMM has reach the mean evaluation 69. The mobiles strategy results in a tracking of the target. The figure 10 illustrates a short sequence of escape/tracking of the target. It has been noticed two quite distinct behaviours, among the many runs of the policy:

- *The two mobiles may both cooperate on tracking the target,*

- *When the target is near the border, one mobile may stay along the opposite border while the other mobile may perform the tracking.* This strategy seems strange at first sight. But it is recalled that the moving

rule of the target tends to neglect a nearby mobile compared to a distant mobile. In this strategy, the first mobile is just annihilating the ability of the target to escape from the tracking of the second mobile.

6   Conclusion

In this paper, we proposed a general method for approximating the optimal planning in a partially observable control problem. Hierarchical HMM families have been used for approximating the optimal decision tree, and the approximation has been optimized by means of the Cross-Entropy method. Moreover, hierarchical HMM has been characterized as a multilevel HMM with alternately up-and-down (Markovian) diffusions of the information between the levels. The algorithm has been applied on a simplier model based on a weaker diffusion of the information.

At this time, the method has been applied to a strictly discrete-state problem and has been seen to work properly. The convergence seems global: the many runs of the algorithm have reach the approximately same optimal values. The optimal HHMMs are able to "track" the target. An interesting point is that these HHMMs have discovered two quite different global strategies and are able to choose between them: make the mobiles both cooperate on tracking or require one mobile for deluding the target.

The results are promising, but the example test is still simple. The observation and action spaces are limited to a few number of states. And what happens if the hidden space becomes much more intricated? There are several possible answers to such difficulties:

First, it has been shown that it is not necessary to define the priors of the problem (defining priors often implies many approximations): the CE algorithm is able to learn the policy directly from the actual world. Moreover, the cross-entropic principle could be applied for optimizing continuous laws. It is thus certainly possible to consider mixed continuous/discrete HHMM, which are more realistic for a planning policy. At last, many refinements are foreseeable about the structure of the HHMMs. More precisely, hierarchic models for observation, action and memory (hierarchical HMMs, but also

hierarchical Markov Random Fields) should be improved in order to locally factorize intricated problems. This research is just preliminary and future works should investigate these questions.

References

[1] De Boer and Kroesse and Mannor and Rubinstein, *A Tutorial on the Cross-Entropy Method*,
http://www.cs.utwente.nl/~ptdeboer/ce/

[2] Richard Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

[3] Edward J. Sondik, *The Optimal Control of Partially Observable Markov Processes*, PhD thesis, Stanford University, Stanford, California, 1971.

[4] Anthony Rocco Cassandra, *Exact and approximate algorithms for partially observable Markov decision processes*, PhD thesis, Brown University, Rhode Island, Providence, May 1998.

[5] Georgios Theocharous, *Hierarchical Learning and Planning in Partially Observable Markov Decision Processes*, PhD thesis, Michigan State University, 2002.

[6] Shai Fine and Yoram Singer and Naftali Tishby, *The Hierarchical Hidden Markov Model: Analysis and Application*, Machine Learning, 1998.

[7] Kevin Murphy and Mark Paskin, *Linear Time Inference in Hierarchical HMMs*, Proceedings of Neural Information Processing Systems, 2001.

[8] Homem-de-Mello, Rubinstein, *Rare Event Estimation for Static Models via Cross-Entropy and Importance Sampling*,
http://users.iems.nwu.edu/~tito/list.htm

A   Murphy and Paskin Bayesian Network

Definition.   A HHMM is a HMM which output is either an observation (when reaching the output level) or a HHMM. A formal definition is given

here, for peoples acquainted with such HMM formalism. But it is not explained in detail.

Formalism. A HHMM with Δ levels is characterized by an observation set Q_1 and by $\Delta - 1$ quadruplet $(Q_d, E_d, \rho_d, \pi_d)_{2 \leq d \leq \Delta}$. For each level $d \in [\![2, \Delta]\!]$, the quadruplet $(Q_d, E_d, \rho_d, \pi_d)$ is related to a HMM for the level d, thus verifying:

- Q_d is a set of states for the level d,

- $E_d \subset Q_d$ is a set of ending states (may be empty for the root Δ). When such state is obtained, the HMM stops and jumps back to the HMM of higher level; for the root level, the HHMM just ends,

- $\rho_d(q_{d-1,t}, q_{d,t}) = P(q_{d-1,t}|q_{d,t})$ describes the probability for the HMM at level d to produce (at time t) the output $q_{d-1,t} \in Q_{d-1}$ when the inner state is $q_{d,t} \in Q_d$. When $d = 2$, the output is just an observation produced by the HHMM. When $d > 2$, the HMM of level d initiates the children HMM of level $d - 1$ with the starting state $q_{d-1,t}$, and waits until the children HMM reachs an ending state,

- $\pi_d(q_{d,t}, q_{d,t+1}) = P(q_{d,t+1}|q_{d,t})$ describes the probability for the HMM at level d to transit from a state $q_{d,t} \in Q_d \setminus E_d$ at time t to the state $q_{d,t+1} \in Q_d$ at time $t + 1$. This transit is runned *after* the production stage.

In this definition, $\pi_\Delta(\emptyset, q_{\Delta,1}) = P(q_{\Delta,1})$ initializes the HHMM at time 1.

Related Bayesian Network. Murphin and Paskin[7] proposed an alternate definition of the HHMM by means of Bayesian Network. This BN is in fact a vectorized HMM, which has the Markov property both in time and in hierarchical level. This BN is relying on 2 type of cells: state cells (symbols $\bullet, \triangledown$) and boolean cells (symbols $\oslash, \odot, \otimes$). More precisely:

$\bullet$ means an inner state cell,

$\triangledown$ means an output state cell,

$\oslash$ means a *unspecified* boolean cell,

$\odot$ means a boolean cell *specified* FALSE,

$\otimes$ means a boolean cell *specified* TRUE.

Figure 11: A Hierarchical HMM (4 levels)
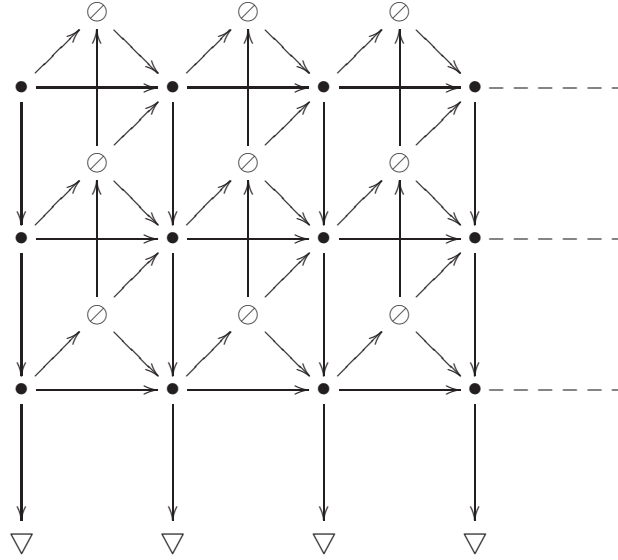
The structure of the Bayesian Network is given by figure 11. The boolean cells are dedicated to the control of the state transition. A $\odot$/FALSE cell indicates that the bottom level is running. A $\otimes$/TRUE cell indicates that the bottom level is ended. Depending on the boolean configuration, it is chosen a HMM transit π, a HMM production ρ or a wait (the states are leaved unchanged by the identity id). All the state transitions are summarized in tables 1 and 2. For example, when a children level is running, the father level is waiting and the father row is between two $\odot$ cells: the transition is id and the father state is leaved unchanged

On the other hand, the table 3 summarizes the boolean transitions. A boolean cell may just be a copy of the bottom boolean cell, when this cell is False. Indeed, that means that the children level is running, and consequently the father level is running although in a waiting mode. Otherwise, the transition is controlled by the HMM transit π.

Bayesian Network equivalence. It is shown now that the BN of figure 11 is a particular case of the BN of figure 7. The simple hierarchical model of figure 7 is thus sufficient to describes a general HHMM.
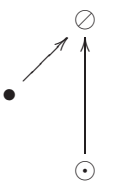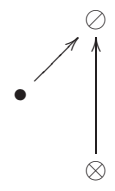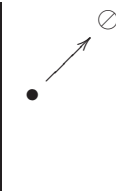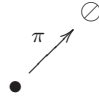
28

Table 1: Transition rule: for a state cell

Table 2: Transition rule: for a state cell (border cases)

Table 3: Transition rule: for a boolean cell

The BN of Murphy is really not far from figure 7. It is just needed to distort the arrows of figure 11, to add intermediate state cells, denoted $\star$, and intermediate boolean cells, denoted $\boxminus$. The intermediate Bayesian Network of figure 12 is obtained then, and is clearly equivalent to the BN of Murphy. By fusing the neighbooring cells $\oslash, \star, \boxminus$ into the cells $\circ$ and redefining the transition rules properly, the BN of figure 13 is resulting, which is of the same kind as in figure 7. Notice that the cells $\circ$ are possibly containing two booleans and an information.
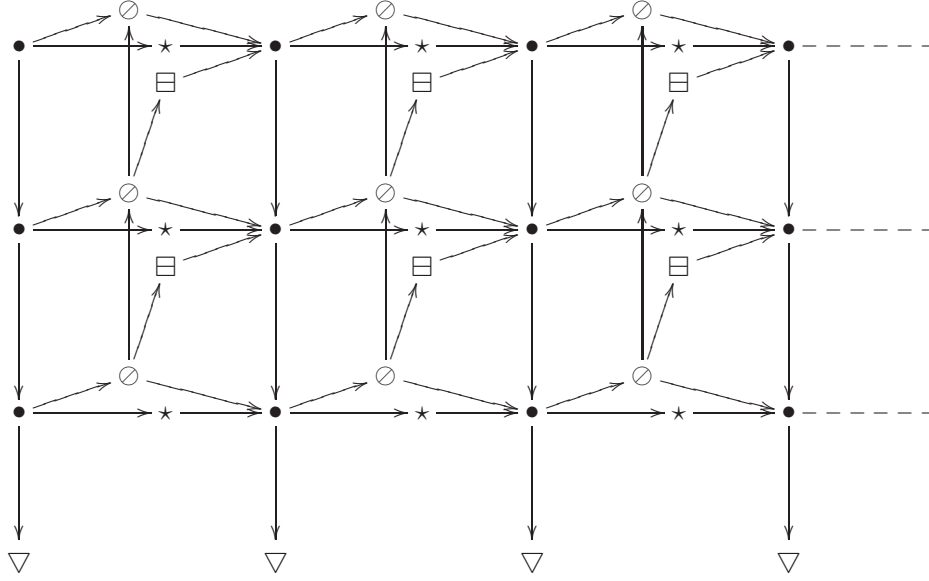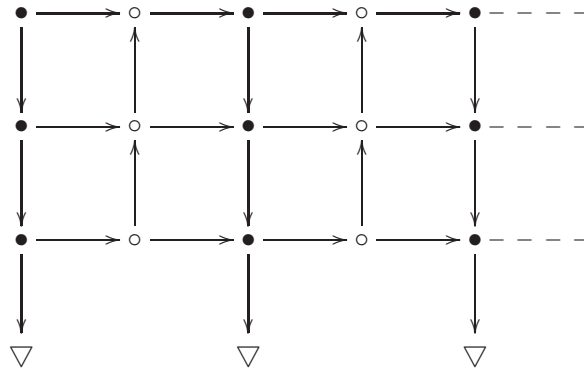
Figure 12: Modified Hierarchical HMM

Figure 13: Simplified Model of a HHMM

○ contains ⋆ , ⊘ and possibly ⊟